Mecánica Computacional Vol. XXIII, pp. 3333-3349
G.Buscaglia, E.Dari, O.Zamonsky (Eds.)
Bariloche, Argentina, November 2004

VISUALIZAÇÃO DE RESULTADOS DE SIMULAÇÕES COMPUTACIONAIS ATRAVÉS DA REALIDADE VIRTUAL

José Luís Farinatti Aymone *

*Núcleo de Computação Gráfica Aplicada, Departamento de Expressão Gráfica
Faculdade de Arquitetura, Universidade Federal do Rio Grande do Sul
Rua Osvaldo Aranha, 99 sala 408, Porto Alegre, CEP: 90035-190 RS, Brasil
e-mail: aymone@ufrgs.br, web page: http://www.campusvirtual.ufrgs.br

Key words: Virtual Reality, Scientific Visualization, Computational Simulation, Finite Element Method

Abstract. *The visualization of results obtained with numerical analysis using methods such as the Finite Element Method is a problem that has been faced since the computational simulation is used in engineering. Post-processing of 2D or 3D geometry, vector (as velocities) and scalar (as stresses) variables is performed using features as colors according to the value, arrows indicating direction, mesh visualization, rotation and pan. To employ the virtual reality for post-processing the results, the user should write in the numeric analysis code an output routine in VRML (Virtual Reality Modeling Language) format. The VRML visualization is executed in internet browsers by installing free VRML navigation plug-ins, without needing commercial softwares. This work presents details of VRML language to visualize scalar variables and to make animation. Engineering examples and the software of scientific visualization in virtual reality in development are shown. An advantage of VRML is that any user without deep computer graphics knowledge is able to use it to visualize the results of numerical analysis.*

1 INTRODUÇÃO

A realidade virtual é uma tecnologia que pode ser empregada em diversas áreas, como educação (ensino a distância), medicina (estudo do corpo humano), arquitetura (prédios e interiores) e engenharia (peças e simulação numérica). Ela permite a interação com o usuário e a navegação pelo ambiente virtual em tempo real.

A linguagem VRML -Virtual Reality Modeling Language-, ou Linguagem de Modelamento da Realidade Virtual[1], é um dos formatos mais utilizados e alia qualidade visual com velocidade de navegação e fácil disponibilização através da Internet.

Pesquisas sobre a linguagem VRML vêm sendo desenvolvidas desde 2001. O seu resultado são modelos em realidade virtual dos prédios do Campus Central da Universidade Federal do Rio Grande do Sul[2,3] disponíveis para navegação no site *www.campusvirtual.ufrgs.br* (Figuras 1 e 2).

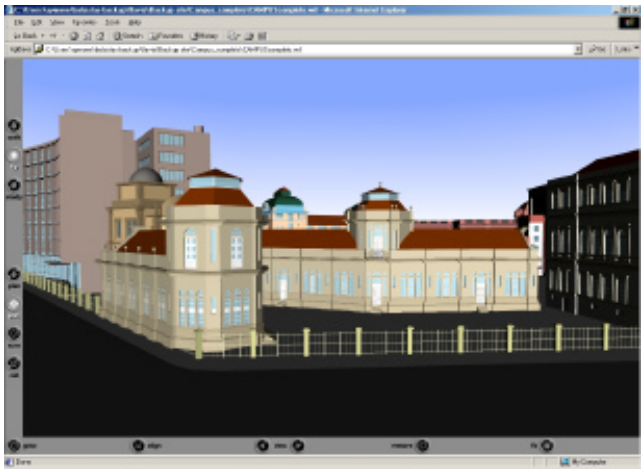

Figura 1- Campus Central da UFRGS em realidade virtual

Figura 2- Saguão da Faculdade de Arquitetura da UFRGS em realidade virtual

Nas Figuras 1 e 2 mostra-se uma tela do Internet Explorer com o plug-in Cortona VRML[4] instalado, no qual os botões à esquerda e abaixo na janela permitem a navegação em realidade virtual.

O processo de transformação dos modelos estáticos em modelos interativos é realizado em três etapas[2]. Os prédios são desenvolvidos individualmente no AutoCAD (formato *.dwg*). Como o AutoCAD[5] (Aymone e Teixeira, 2002) não realiza a exportação para VRML, os arquivos *.dwg* são exportados com o formato *.3ds* para o 3DStudio. No 3DStudio[6] (Peterson, 1998) são aplicados materiais, texturas e animações de movimento. Em seguida, os arquivos são exportados para VRML (formato *.wrl*). A navegação em VRML é feita no Internet Explorer, utilizando plug-ins gratuitos como o Cortona VRML 4.0.

O Cortona VRML 4.0 foi escolhido por apresentar boa velocidade de navegação e excelente qualidade de visualização, reproduzindo fielmente o espaço real e possibilitando a interatividade com o usuário.

A linguagem VRML permite desenhar objetos diretamente, utilizando seus comandos em um arquivo texto com extensão *.wrl* . Investigando esta linguagem, observou-se que ela possui as funcionalidades necessárias à visualização científica de resultados de análises numéricas realizadas através de métodos como o Método dos Elementos Finitos[7,8]. Pode-se citar as seguintes funcionalidades: visualização da geometria de peças ou estruturas bi e tridimensionais e dos resultados, variáveis escalares (como tensões) e vetoriais (como velocidade e aceleração) obtidas quando se submete a estrutura a cargas, esforços, fadiga, conformação mecânica, efeitos de temperatura, etc. As variáveis escalares podem ser mostradas através de contornos com cores conforme o seu valor e as variáveis vetoriais podem ser vistas através de setas que representam a direção do movimento. Além disso, em análises dinâmicas e de grandes deformações[9] são utilizadas animações, onde se pode ver as etapas do processo de fabricação de uma peça [10] ou o impacto de um automóvel contra uma parede. A geometria é vista em 2D ou 3D, através de uma cor ou da malha (nós e elementos) que compõe a estrutura. O usuário interage em tempo real, rotacionando, aproximando, movendo e podendo identificar o número dos nós e dos elementos.

Nas próximas seções são detalhados os procedimentos para a conversão em realidade virtual dos resultados de análises numéricas e apresentadas as funcionalidades implementadas em um software de visualização científica, em desenvolvimento, que utiliza a realidade virtual.

2 GERAÇÃO DO ARQUIVO EM REALIDADE VIRTUAL A PARTIR DOS RESULTADOS DA ANÁLISE NUMÉRICA

Nesta seção apresenta-se o formato VRML para visualização da geometria (malha e elementos), variáveis escalares (como tensões) e animações.

2.1 Dados da geometria

Os dados geométricos de saída de um programa de análise numérica que utiliza o métodos dos elementos finitos normalmente consistem em coordenadas (x, y e z) dos nós e conetividades dos elementos (nós que compõem os elementos). A seguir apresenta-se o

formato VRML da geometria de uma peça.

A primeira linha do arquivo *.wrl* deve ter o comando da Figura 3 na primeira linha para que seja interpretado como arquivo VRML.

```
#VRML V2.0 utf8
```

Figura 3- Primeira linha do arquivo *.wrl*

No VRML, deve-se definir separadamente as linhas da malha e o preenchimento com uma cor das faces dos elementos.

Na Figura 4, define-se as linhas da malha. As observações nas caixas de texto à direita estão detalhadas abaixo da Figura 4.

```
DEF Malha1 Transform {
 children [ Shape {
 appearance Appearance {
    material Material {
    diffuseColor  0.00E+00  0.00E+00  0.00E+00
    emissiveColor  0.00E+00  0.00E+00  0.00E+00
     }}

 geometry IndexedLineSet {

    coord DEF Malha Coordinate {
           point [
  0.00000E+0000  0.00000E+0000  0.00000E+0000,
  7.34680E+0000  0.00000E+0000  0.00000E+0000,
  1.46680E+0001  0.00000E+0000  0.00000E+0000,
           ]
    }
    coordIndex [
 0,1,65,64 -1
 0,1,9,8 -1
 0,8,72,64 -1
 1,2,66,65 -1
 441,442,631,624 -1
    ]
} } ] }
```

(1) Malha com cor no formato RGB da linha da malha (entre 0 e 1)

`(2)IndexedLineSet:` Permite definir os nós e linhas que os unem

(3) Coordenadas `x, y e z` dos nós

(4) Conetividades das faces quadriláteras. O primeiro nó deve ter nome `0`. O valor `-1` separa uma face de outra.

Figura 4- Comandos para desenhar a malha

Observações da Figura 4

(1) A cor em componentes RGB (Red-Green-Blue ou Vermelho-Verde-Azul) é bastante utilizada e pode ser encontrada tanto em programas gráficos como editores de texto. A mistura dos valores entre 0 e 1 no caso do VRML, ou entre 0 e 255, no caso de alguns outros programas permite obter as cores desejadas. Caso se conheça as componentes da cor desejada entre 0 e 255, basta dividir o seu valor por 255 e obter o valor no intervalo [0;1].

(2) A geometria da malha é definida no comando `IndexedLineSet { }`. Utilizando a cor escolhida acima, são desenhadas linhas entre os nós, que mostram a malha.

(3) As coordenadas x, y z dos nós são colocadas na opção `point [ ]`

(4) As conetividades das faces dos elementos (no caso acima o elemento pode ser um quadrilátero de 4 nós ou hexaedro de 8 nós, que possui 4 nós na face) são definidas na opção `coordIndex [ ]`, sendo que o valor `-1` separa as faces.

A Figura 5 apresenta a malha em VRML do problema do puncionamento hemisférico (puncionamento de um cilindro metálico por uma matriz de conformação rígida hemisférica) utilizando elementos hexaédricos de 8 nós. Foi utilizado na análise o código de elementos finitos METAFOR 3D[11,12].

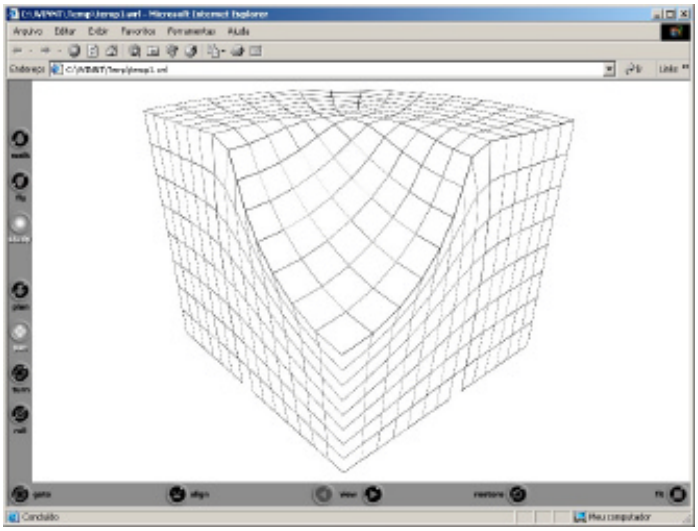

Figura 5- Puncionamento hemisférico. Elementos hexaédricos de 8 nós.

Para a visualização das faces sombreadas com cor, é necessário um procedimento similar ao descrito na Figura 4, mas utilizando o comando `IndexedFaceSet` para colocar cor nas faces, apresentado na Figura. 6.

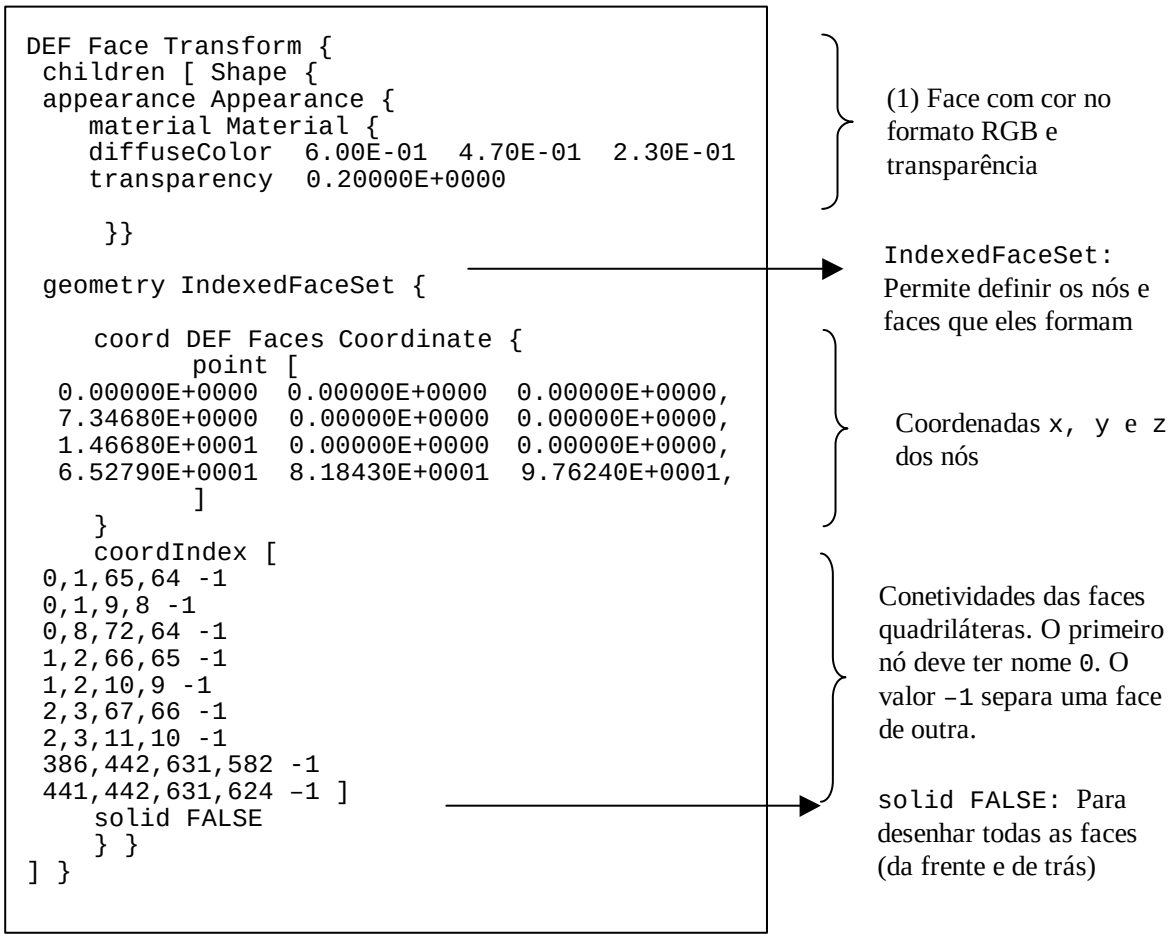

Figura 6- Comandos para desenhar com cor as faces dos elementos

Observações da Figura 6

(1) A face tem uma cor em componentes RGB e pode ter transparência para permitir a visualização do interior da peça. A transparência varia de zero (sem transparência) a 1 (totalmente transparente).

A Figura 7 apresenta o problema da Barra de Taylor[12] (projetar um cilindro metálico, com uma velocidade inicial, contra uma superfície rígida) com as faces sombreadas com cor.

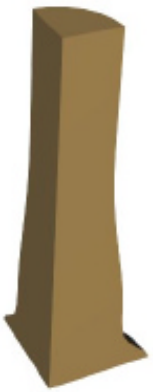

Figura 7- Barra de Taylor utilizando elementos hexaédricos de 8 nós

As seções 2.2 e 2.3 apresentam o procedimento para a visualização de resultados escalares e vetoriais associados à análise.

2.2 Variáveis escalares

Variáveis escalares, tais como tensões, são visualizadas através de um espectro de cores no qual os elementos recebem cor conforme o valor da tensão neles existente. O espectro de cores é definido por uma paleta, onde normalmente se utiliza o azul para os valores mais baixos e o vermelho para os mais altos, havendo uma interpolação das cores nas faces, a partir do valor nos nós.

A linguagem VRML permite a interpolação de cores através do comando `Color`, que deve ser colocado logo após o comando `solid FALSE` da Figura 6. O comando `Color` permite atribuir uma cor no formato RGB para cada nó da malha. Na visualização em realidade virtual, a interpolação da cor entre os nós é feita automaticamente, seguindo do azul em RGB (0 0 1) para o verde (0 1 0) e depois para o vermelho (1 0 0).

Deve-se atribuir cor azul ao menor valor da variável escalar, vermelho para o maior valor e obter os valores intermediários por regra de três. A Figura 8 apresenta o comando Color.

```
...continuação da Figura 6
   726,735,760,745 -1
   745,747,775,760 -1
      ]
      solid FALSE

      color Color {
      color [
    1.00000E+0000                    0.00000E+0000
0.00000E+0000,
    1.00000E+0000                    7.50000E-0001
0.00000E+0000,
    1.00000E+0000                    1.00000E+0000
0.00000E+0000,
            ] }
      } } ] }
```

Conetividades e `solid FALSE:` da Figura 6

Comando `Color` com uma cor RGB por nó em cada linha

Figura 8- Comando para a visualização de variáveis escalares com cores

A Figura 9 mostra o resultado da análise do problema de hidroconformação (método para a manufatura de componentes que alia, às tradicionais matrizes de conformação, o uso da água sob pressão) de um tubo em T^{10} com valores de deformação plástica equivalente. Os valores de deformação plástica equivalente (variável escalar) são apresentados com cores conforme o seu valor, os valores máximos estão em vermelho e os mínimos em azul. Há 10% de transparência para a visualização através da peça.

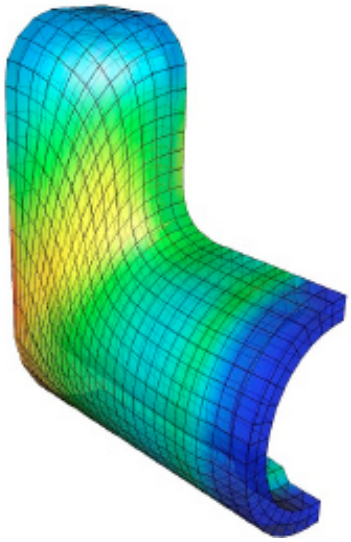

Figura 9- Hidroconformação de um tubo. Elementos hexaédricos de 8 nós.

2.3 Variáveis vetoriais

A visualização de variáveis vetoriais, como velocidade, pode ser feita através de setas indicando a sua direção e magnitude. A Figura 10 apresenta os comandos em VRML para a visualização de resultados vetoriais, inseridos como continuação da Figura 6. A seta é desenhada a partir de três cilindros e posicionada através de translações para as coordenadas dos nós, rotações para a direção do vetor e tamanho conforme o módulo do vetor.

Observações da Figura 10

(1) A geometria da seta é definida por três cilindros em `Vetor`. O `VetorInicial` coloca a seta inicial na posição (0,0,0) (`translation 0 0 0`), que não deve ser desenhada. Por isso, o valor da escala é zero (`scale 0 0 0`). O valor de rotação (`rotation 0 0 1 3.1415`) posiciona a seta inicial na direção negativa de Y.

(2) A seta é posicionada em cada nó, através do valor em (`translation 1.54E+01 0.00E+00 0.00E+00`), rotacionada para a direção do vetor (velocidade por exemplo) com o valor em (`rotation 0.00E+00 0.00E+00 -1.00E+00 1.57E+00`) e com o tamanho definido pelo módulo do vetor colocado na escala (`scale 9.19206E-01 4.32629E-01 9.19206E-01`). Para não repetir o objeto seta em cada nó, o que tornaria o arquivo muito grande, utiliza-se o comando USE, que funciona como um clone, aproveitando a geometria da

seta inicial mas colocando-a na posição correta. Dessa forma, independentemente do número de nós da malha, a linguagem VRML considera como se houvesse apenas uma seta para a visualização em realidade virtual.

```
...continuação da Figura 6
441,442,631,624 –1
726,728,747,745 -1
   ]
   solid FALSE
   } }

DEF VetorInicial Transform {
   translation 0 0 0
   rotation  0 0 1 3.1415
   scale 0 0 0
   children [
       DEF Vetor Transform {
              translation  0 2 0
              children [
                     Shape {
                     appearance Appearance {
                     material Material {
                     diffuseColor  1.00E+00  0.00E+00  0.00E+00
                     } }
              geometry Cylinder { radius 0.05 height 4.0 } }
              DEF    Lado1 Transform {
                     translation  -0.2 1.5 0.0
                     rotation 0 0 1 -0.3925
                     children     DEF Arm2 Shape {
                     appearance Appearance {
                     material Material {
                     diffuseColor  1.00E+00  0.00E+00  0.00E+00
                     } }
              geometry Cylinder { radius      0.05 height  1.0
   } } }
              DEF    Lado2 Transform {
                     translation  0.2 1.5 0.0
                     rotation 0 0 1 0.3925
                     children [USE Arm2 ]
              }
]}]}

Transform {
   translation  7.73860E+00  0.00000E+00  0.00000E+00
   rotation  0.00000E+00  0.00000E+00 -1.00000E+00  1.57080E+00
   scale  9.19206E-01  2.29801E-01  9.19206E-01
children USE Vetor }

Transform {
   translation  1.54200E+01  0.00000E+00  0.00000E+00
   rotation  0.00000E+00  0.00000E+00 -1.00000E+00  1.57080E+00
   scale  9.19206E-01  4.32629E-01  9.19206E-01
children USE Vetor }

...Transform dos vetores de cada nó
```

Conetividades e `solid FALSE`: da Figura 6

(1) **`Vetor`** Definição da geometria da seta por cilindros com cor vermelha

(2) Colocar a seta em cada nó com com o comando USE

Figura 10- Comando para a visualização de variáveis vetoriais.

A Figura 11 mostra o resultado de velocidades para o caso do puncionamento hemisférico.

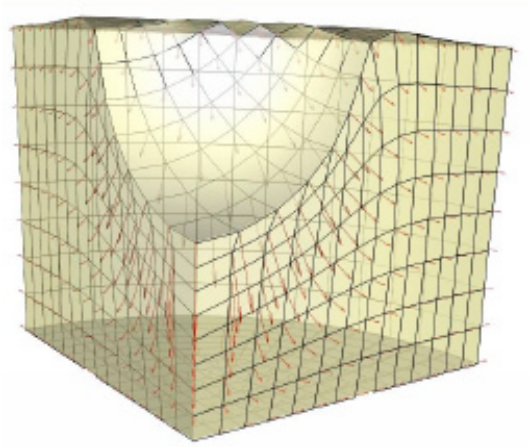

Figura 11- Distribuição de velocidades no puncionamento hemisférico.

2.4 Animações

A realização de animações é feita a partir dos resultados da análise ao longo do tempo. Deve-se ter resultados intermediários da análise gravados em arquivo, um arquivo para cada instante de tempo. Quanto mais arquivos intermediários houver, mais precisa será a animação. Cada arquivo, no formato de realidade virtual *.wrl*, deve conter a geometria e os resultados escalares que se deseja representar, no formato apresentado nas Figuras 6 e 8. Caso se desejar animar apenas a geometria, coloca-se somente os dados da Figura 6.

Para a visualização correta dos resultados escalares, é necessário obter os valores máximos e mínimos da variável entre todos os arquivos. Esses valores são utilizados para realizar a interpolação por cores e visualização da variável, conforme a paleta de cores escolhida. Caso não sejam obtidos os valores máximos e mínimos globais, a interpolação poderá ficar errada pois o valor máximo tende a aumentar ao longo do tempo.

Além disso, deve ser gravado e executado um arquivo *.wrl* como o da Figura 12. Este arquivo gerencia a animação, identificando os arquivos intermediários que a compõe e o tempo de animação. A seguir, detalha-se os comandos da Figura 12.

O comando `Switch { }` em "Choices" permite informar uma lista de arquivos *.wrl* a serem lidos. O valor `whichChoice 0` faz com que a animação comece com o primeiro arquivo da lista em `choice [ ]`.

Cada arquivo *.wrl* intermediário é lido através do comando `Inline` , onde se indica a pasta e o nome do arquivo. Por questão de organização, é interessante numerar os arquivos de resultados em ordem crescente ao longo do tempo (`T001, T002,...`).

O tempo da animação é definido em "Clock" no comando `TimeSensor`. Deve-se colocar o valor de intervalo em `cycleInterval 5`, igual ao número de arquivos intermediários (1 segundo por arquivo - 5 segundos). Com as opções `loop TRUE` e `stopTime<startTime`, a

animação recomeça automaticamente após o último arquivo.

O comando *Script* em "Switcher" define a variação da fração de tempo para a animação. Em *value_changed* deve-se multiplicar a fração *frac* pelo número de arquivos da animação, no caso 5.

As rotas em Route fazem a relação entre os eventos da animação. Na rota ROUTE Clock.fraction_changed TO Switcher.set_fraction, quando há uma variação de tempo em Clock.fraction_changed, essa fração é transmitida para o Switcher.set_fraction, que fará com que o arquivo intermediário seja substituído pelo arquivo seguinte, através da rota ROUTE Switcher.value_changed TO Choices.set_whichChoice, dando a sensação de animação.

```
DEF Choices Switch {
whichChoice 0
choice [
Inline { url "C:\VirtualVIS\punciT001.wrl" }
Inline { url "C:\VirtualVIS\punciT002.wrl" }
Inline { url "C:\VirtualVIS\punciT003.wrl" }
Inline { url "C:\VirtualVIS\punciT004.wrl" }
Inline { url "C:\VirtualVIS\punciT005.wrl" }
]
}

DEF Clock TimeSensor {
cycleInterval 5
loop TRUE
startTime 0.0
stopTime -1.0
}

DEF Switcher Script {
eventIn SFFloat set_fraction
eventOut SFInt32 value_changed
url "vrmlscript:
   function set_fraction( frac, tm ) {
   value_changed = frac * 5;
   }"}

ROUTE Clock.fraction_changed TO Switcher.set_fraction
ROUTE Switcher.value_changed TO
Choices.set_whichChoice
```

Figura 12- Comando para a realização de animações

Na próxima seção apresenta-se o software de visualização científica em realidade virtual que está sendo desenvolvido utilizando os comandos apresentados anteriormente.

3 SOFTWARE DE VISUALIZAÇÃO CIENTÍFICA EM REALIDADE VIRTUAL

O software de visualização científica consiste em duas partes: botões e menus, onde são escolhidas as opções, e janela de visualização em realidade virtual. Esta janela está integrada ao ambiente gráfico e utiliza o plug-in de navegação em realidade virtual Cortona VRML.

Quando o usuário executa alguma ação, por exemplo para visualizar uma componente de tensão, a janela de visualização é atualizada com essa ação. A linguagem de programação do ambiente gráfico é o Delphi[13].

Diversas funcionalidades existentes em softwares comerciais como Tecplot[14] e Gid[15] já estão disponíveis no software de visualização, tais como:

(a) leitura de dados de elementos finitos quadriláteros e hexaédricos.
(b) arquivo de dados (geometria, variáveis escalares e vetoriais) em formato ASCII simples. Os usuários irão inserir no código dos seus softwares de análise numérica, ou em um programa de interface, uma rotina que crie o arquivo de dados no formato pré-definido para possibilitar a visualização científica.
(c) ambiente gráfico tipo Windows, com botões e menus para a realização de operações como: escolha do modo de visualização (wireframe, shade, cor por tensões).
(d) visualização tridimensional em perspectiva e movimentação espacial em 3D dos objetos (rotação, aproximação, ...) em tempo real.
(e) efeitos de iluminação, transparência, sombreamento (*shading*) e eliminação de faces ocultas.
(f) interpolação de cores aplicadas nas faces (para a visualização de tensões) com paleta de cores configurável.

O software de visualização científica gera um arquivo em realidade virtual que pode ser visualizado por usuários que não possuem o ambiente gráfico instalado, mas apenas o browser da internet com o plug-in gratuito de realidade virtual.

A Figura 13 apresenta as etapas para a utilização do software de visualização científica em realidade virtual.

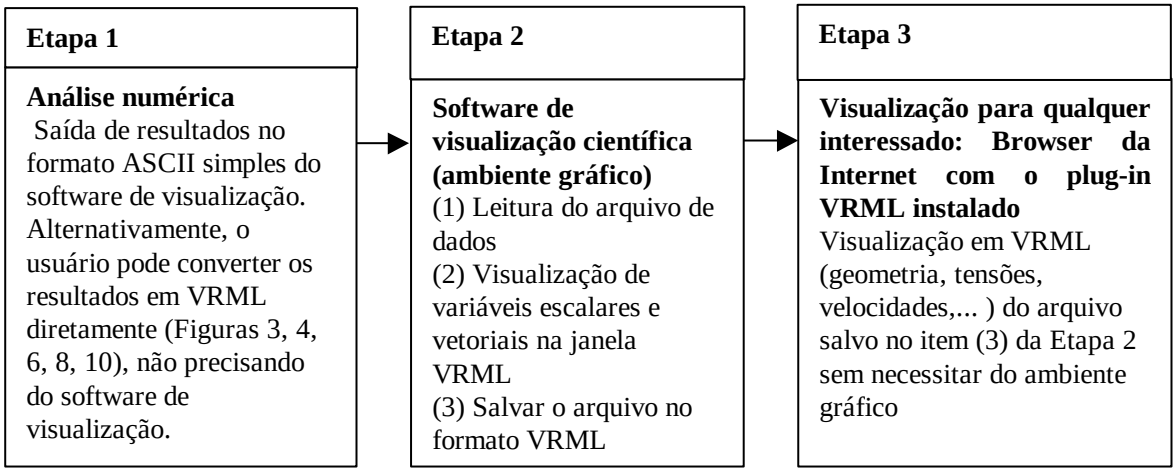

Figura 13- Etapas para para a utilização do software de visualização científica

A Figura 14 apresenta a tela do software de visualização científica em desenvolvimento, chamado VirtualVIS. A janela de visualização em VRML está colocada à direita. É mostrada a variável escalar *epl* (defomação plástica equivalente) com a escala de cores Padrão escolhida pelo usuário para o caso do puncionamento hemisférico.

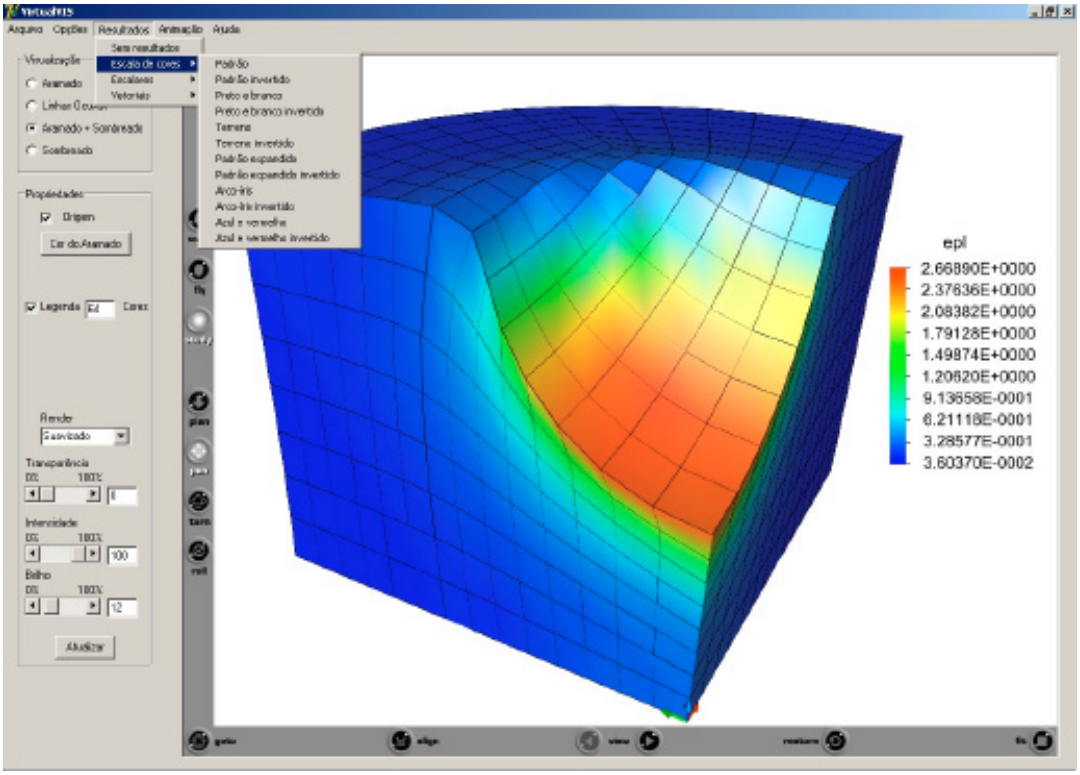

Figura 14- Puncionamento hemisférico: deformação plástica equivalente

O software possui uma interface com fácil acesso aos comandos, com menus na parte superior da tela. Conforme a ação que está sendo executada, as opções disponíveis aparecem na barra de tarefas à esquerda. Já estão implementadas as paletas de cores mais comuns e a legenda para variáveis escalares. O ponto (0,0,0) é identificado por um eixo com cor vermelha (eixo X), verde (eixo Y) e azul (eixo Z). Pode-se escolher o modo de visualização (aramado, linhas ocultas, aramado+sombreado e sombreado), o grau de transparência, a cor da malha e das faces, o tipo de render (suavizado ou facetado) e a cor de fundo da janela.

A Figura 15 mostra o exemplo da barra de Taylor com os resultados de *epl*. Na barra da esquerda da janela do VirtualVIS, aparece o menu de animação, onde se indica como arquivo inicial o primeiro e o final como o oitavo.

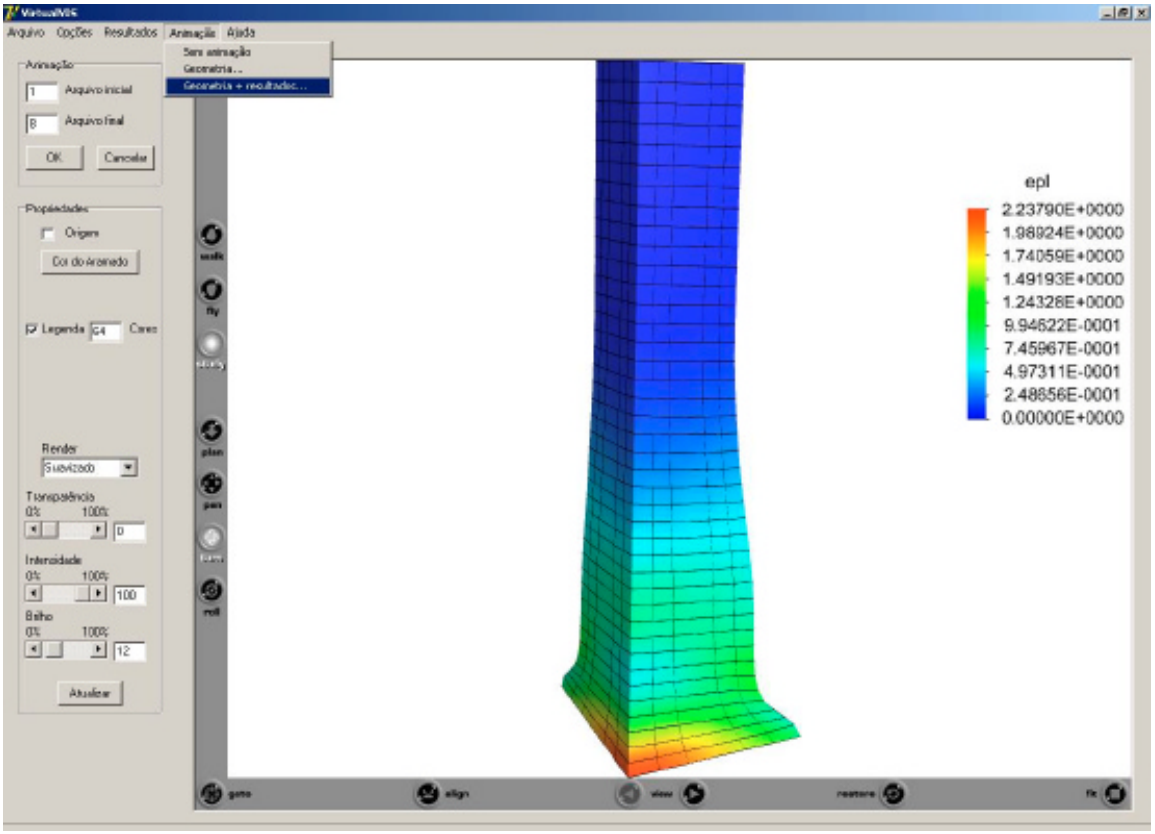

Figura 15- Barra de Taylor: animação da deformação plástica equivalente

A Figura 16 mostra as velocidades para o caso da barra de Taylor. O software possui dois tipos de seta (2D e 3D), seta com tamanho fixo ou variável conforme o módulo da velocidade e fator de escala para a seta. A Figura 16 apresenta o vetor 2D com tamanho fixo e fator de escala 4.

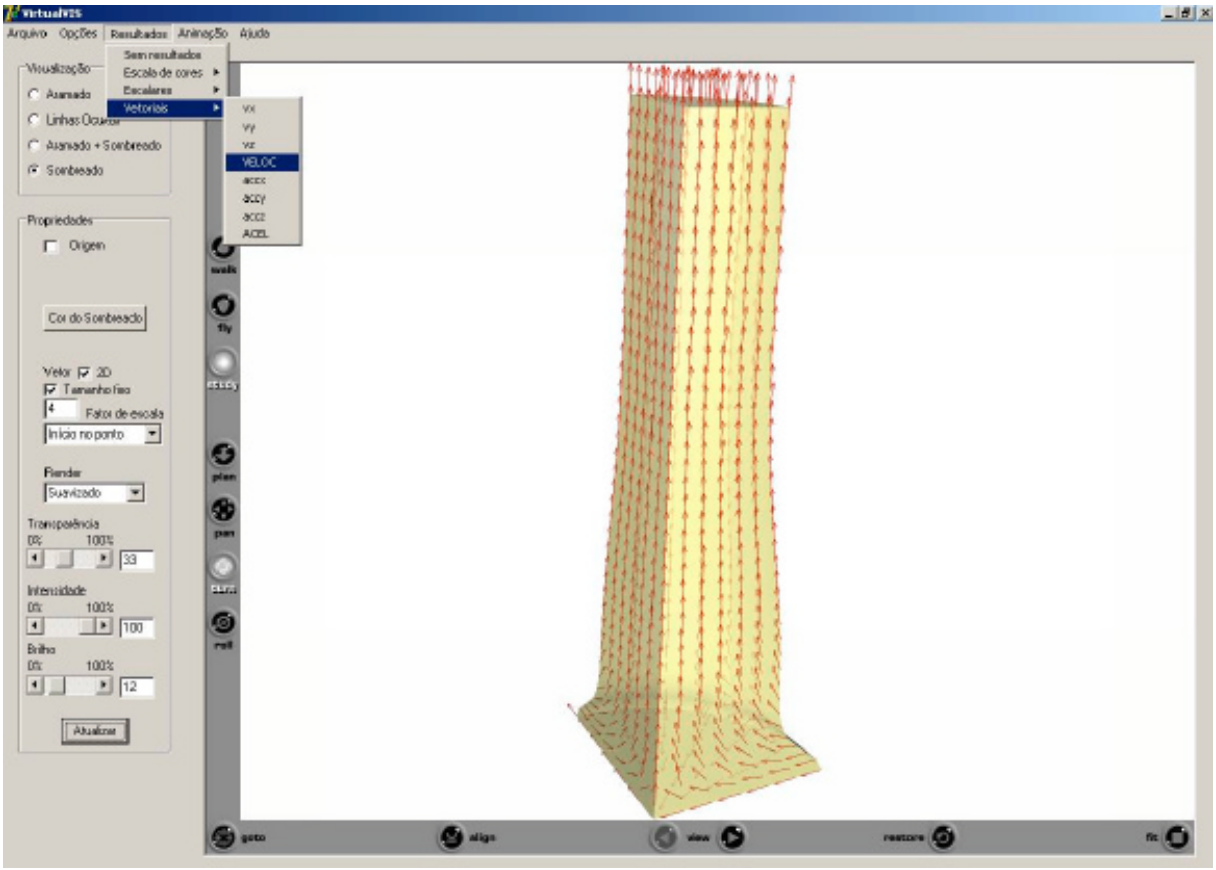

Figura 16- Barra de Taylor: velocidades

3.1 Velocidade de processamento do software

Foram testados os exemplos do puncionamento hemisférico e da barra de Taylor para verificar a performance do software. Os tempos foram medidos em um computador Pentium 4 1.5 GHz com mémoria RAM de 512 Mb.

Na Tabela 1 apresenta-se os tempos para carregar os exemplos. A navegação em todos eles é instantânea, por exemplo, ao se fazer uma rotação, o movimento é em tempo real.

Tabela 1- Tempos para carregar os exemplos

Exemplo	Nós	Elementos	Arquivo escalar+vetor (Kb)	Tempo (s)	Animações	
					Nº de arquivos	Tempo (s)
Puncionamento	1.176	1.521	247	< 1	8	7
Barra de Taylor	1.080	1.517	303	< 1	20	9

Da Tabela 1, observa-se que o tempo para carregar as animações é proporcionalmente mais

alto, porque é necessário transformar cada arquivo de um formato padrão para o formato VRML (ler o arquivo e salvá-lo em VRML) para depois realizar a animação.

Uma vez que não se possui, no momento, exemplos de elementos finitos de grande porte no formato necessário para visualização em VRML, para avaliar a velocidade de processamento vai se utilizar os modelos em VRML do Campus Virtual de UFRGS (disponíveis no endereço www.campusvirtual.ufrgs.br), que possuem dezenas de milhares de elementos e podem apresentar um valor aproximado do tempo necessário para carregar e navegar em exemplos de elementos finitos de mesma escala de grandeza. Em VRML, cada elemento hexaédrico é subdividido em 6 faces quadriláteras, uma vez que o VRML apresenta apenas elementos triangulares e quadriláteros. Nos exemplos dos prédios, são utilizadas faces triangulares.

Tabela 1- Tempos para carregar os prédios do Campus da UFRGS

Prédio	Nº de faces	Tamanho (Mb)	Tempo (s)
Biociências	76.839	2,400	1,0
Campus completo (25 prédios)	1.000.474	11,1	20,0

No Campus Completo, devido ao elevado número de faces, a navegação (rotação por exemplo) não acontece instantaneamente, mas leva poucos segundos, tornando aparentemente viável a utilização da linguagem VRML em grandes problemas de elementos finitos.

4 CONCLUSÕES

A linguagem VRML se mostra apropriada à visualização científica de resultados, uma vez que alia as funcionalidades necessárias à visualização da geometria e dos resultados com a excelente qualidade gráfica da realidade virtual.

A conversão dos resultados para VRML não necessita de conhecimentos aprofundados de computação gráfica, apenas conhecimentos básicos de VRML.

Ao converter os resultados da análise numérica diretamente em VRML, qualquer usuário têm acesso à visualização, visto que o plug-in VRML é gratuito e disponível na internet.

O software de visualização científica facilita a obtenção da visualização utilizando diversas opções, tais como: linhas invisíveis, sombreamento com cores escolhidas pelo usuário, transparência, diferentes paletas de cores, eixo na origem do sistema, escolha das variáveis escalares e vetoriais a serem exibidas através de menus, etc.

5 REFERÊNCIAS

[1] J. Hartman, and J. Wernecke, *The VRML 2.0 Handbook*, Silicon Graphics, New York, (1996).

[2] J. L. F. Aymone, L. B. Kochenborger, R. B. Trindade and B. B. Soriano, “A Realidade Virtual Aplicada ao Ensino de Engenharia”, *XXX Congresso Brasileiro de Ensino de Engenharia (XXX COBENGE)*, Piracicaba, CD-ROM (2002).

[3] J. L. F. Aymone, M. Fensterseifer and T. Garbachi, "A Realidade Virtual Aplicada à Arquitetura e Engenharia", *V International Conference on Graphics Engineering for Arts and Design (GRAPHICA 2003)*, Santa Cruz do Sul/RS, CD-ROM (2003).
[4] Parallel Graphics,. *Cortona VRML Client 4.1*, (2003) http://www.parallelgraphics.com/products/cortona/
[5] J. L. F. Aymone and F. G. Teixeira. *AutoCAD 3D Modelamento e Rendering*, Artliber Editora, São Paulo, (2002).
[6] M. T. Peterson, *Fundamentos do 3D Studio MAX*, Editora Campos, Rio de Janeiro, (1998).
[7] T. J. R. Hughes, *The Finite element method*. Prentice-Hall, (1987).
[8] K-J. Bathe. *Finite element procedures*, Prentice-Hall, (1996).
[9] R. M. McMeking and J. R. Rice, "FE formulation for problems of large elastic-plastic deformation", *International Journal of Solids and Structures*, **11**, 601-616 (1975).
[10] J. L. F. Aymone, F. G. Teixeira, E. Bittencourt and G. J. Creus, "Numerical simulation of the hydroforming process", *9º Congresso e Exposição Internacionais de Tecnologia da Mobilidade (SAE BRASIL 2000)*, São Paulo (2000).
[11] J. L. F. Aymone, E. Bittencourt and G. J. Creus, "Simulation of 3-D Metal-Forming Using an Arbitrary Lagrangian-Eulerian Finite Element Method", *Journal of Materials Processing Technology*, **110**, 218 – 232 (2001).
[12] J. L. F. Aymone, "Mesh motion techniques for the ALE formulation in 3-D large deformation problems", *International Journal for Numerical Methods in Engineering*, **59**, 1879-1908 (2004).
[13] Borland,. *DELPHI 7 Studio*, (2002) http://www.borland.com/delphi/
[14] Amtec Engineering, *TECPLOT 10*, 2003, http://www.amtec.com/products/tecplot/tecplot_main.htm.
[15] CIMNE,. *GID 7.0*, (2003) http://gid.cimne.upc.es/